Intelligent Agents

Pınar Yolum
p.yolum@uu.nl

Utrecht University
Web Ontology Language

Based largely on
Web Ontology Language (OWL)

• RDF captures the basics, i.e., an object-oriented type system
• Additional subtleties of meaning are needed for effective KR
• OWL builds on RDF
OWL in Brief

• Specifies classes and properties in a form of description logic (DL)
  – Class operators analogous to Boolean operators \textit{and, not, and or}
  – Constraints on properties: transitive, …
  – Restrictions: constructs unique to DL

• OWL 1.0 has three species: OWL Full, OWL DL, and OWL Lite

• OWL 2.0
Constructing OWL Classes

• Assertions (Specify the class in relation to other classes)

```xml
<owl:Class rdf:about="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#Dessert">
  <rdfs:subClassOf rdf:resource="20031209/food#EdibleThing"/>
  <owl:disjointWith rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#OtherTomatoBasedFood"/>
</owl:Class>
```
<owl:Class rdf:about="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#Fruit">
  <owl:equivalentClass>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#NonSweetFruit"/>
        <rdf:Description rdf:about="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#SweetFruit"/>
      </owl:unionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:disjointWith rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#OtherTomatoBasedFood"/>
</owl:Class>
Set operators

- intersectionOf, unionOf, complementOf

```xml
<owl:Class rdf:ID="Fruit">
  <owl:unionOf rdf:parseType='Collection'>
    <owl:Class rdf:resource="#SweetFruit" />
    <owl:Class rdf:resource="#NonSweetFruit" />
  </owl:unionOf>
</owl:Class>

<owl:Class rdf:ID='SugaryBread'>
  <owl:intersectionOf rdf:parseType='Collection'>
    <owl:Class rdf:about='#Bread'/>
    <owl:Class rdf:about='#SweetFood'/>
  </owl:intersectionOf>
</owl:Class>

<owl:Class>
  <owl:complementOf>
    <owl:Class rdf:about="#Meat"/>
  </owl:complementOf>
</owl:Class>
```
Enumeration

• List instances that are members of a class

<owl:Class rdf:ID="Continent">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Eurasia"/>
    <owl:Thing rdf:about="#Africa"/>
    <owl:Thing rdf:about="#NorthAmerica"/>
    <owl:Thing rdf:about="#SouthAmerica"/>
    <owl:Thing rdf:about="#Australia"/>
    <owl:Thing rdf:about="#Antarctica"/>
  </owl:oneOf>
</owl:Class>
OWL Entities and Relationships

- rdfs:Class
  - rdfs:subClassOf
  - owl:equivalentClass
  - owl:disjointWith
  - rdf:domain
  - owl:inverseOf
  - owl:equivalentProperty
  - rdfs:subPropertyOf
  - rdf:range

- rdfs:Datatype
- owl:DataRange
- owl:Datatype Property
- owl:Object Property
- owl:Inverse Functional Property
- owl:Symmetric Property
- owl:Transitive Property
- rdf:Property
- owl:Functional Property
- owl:Inverse Functional Property

Spring 2018 — Pınar Yolum
OWL Object Properties

• Transitive Property
  – P(x,y) and P(y,z) implies P(x, z)

<owl:ObjectProperty rdf:ID="locatedIn">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="&owl;Thing" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>

<Region rdf:ID="SantaCruzMountainsRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
</Region>

<Region rdf:ID="CaliforniaRegion">
  <locatedIn rdf:resource="#USRegion" />
</Region>
OWL Object Properties

• Symmetric Property
  – $P(x,y)$ iff $P(y,x)$

```xml
.owl:ObjectProperty rdf:ID="adjacentRegion">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:domain rdf:resource="#Region" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>

<Region rdf:ID="MendocinoRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
  <adjacentRegion rdf:resource="#SonomaRegion" />
</Region>
```
OWL Object Properties

• Functional Property
  – P(x, y) and P(x,z) implies y=z

<owl:Class rdf:ID="VintageYear" />
<owl:ObjectProperty rdf:ID="hasVintageYear">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Vintage" />
  <rdfs:range rdf:resource="#VintageYear" />
</owl:ObjectProperty>
OWL Object Properties

- inverseOf
  - P1(x,y) iff P2(y,x)

```xml
<owl:ObjectProperty rdf:ID="hasMaker">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="producesWine">
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
```
OWL Object Properties

• inverseFunctionalProperty
  – P(y,x) and P(z,x) implies y=z

<owl:ObjectProperty rdf:ID="hasMaker" />
<owl:ObjectProperty rdf:ID="producesWine">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
Restrictions: 1

• A unique feature of description logics
• Like division: define classes in terms of a restriction that they satisfy with respect to a given property
• Anonymous: typically included in a class def to enable referring them
• Key primitives are
  – someValuesFrom a specified class
  – allValuesFrom a specified class
  – hasValue equal to a specified individual or data type
  – minCardinality
  – maxCardinality
  – Cardinality (when maxCardinality equals minCardinality)
The maker of a Wine must be a Winery. The allValuesFrom restriction is on the hasMaker property of this Wine class only. Makers of Cheese are not constrained by this local restriction.
A player is anyone who plays for some AllStarTeam.

<owl:Class rdf:ID="Player">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="playsFor" />
      <owl:someValuesFrom rdf:resource="#AllStarTeam"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
Restrictions: 4

- **Cardinality**: Ex: Things that have exactly nine players

  `<owl:Restriction>`
  `<owl:onProperty rdf:resource="#hasPlayer" />`
  `<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">9</owl:cardinality>`

  `</owl:Restriction>`

- **MinCardinality or maxCardinality**

  `<owl:Class rdf:about="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#Juice">`
  `<rdfs:subClassOf>`
   `<owl:Restriction>`
   `<owl:onProperty rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#madeFromFruit"/>`
   `<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:minCardinality>`
   `</owl:Restriction>`
  `</rdfs:subClassOf>`
  `</owl:Class>`
Inference

• OWL is about content, not the syntax
• Statements from different documents about the same URI are automatically conjoined
• OWL declarations may seem conflicting
  – Declare that no one can have more than one mother
  – Declare Mary is John’s mother
  – Declare Jane is John’s mother
• A DBMS would declare an integrity violation
• An OWL reasoner would say Mary = Jane
Resources

• Wine ontology: http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf
• Food ontology: http://www.w3.org/TR/2004/REC-owl-guide-20040210/food.rdf
• W3C: http://www.w3.org/TR/owl2-overview/
• Editor: Protégé http://protege.stanford.edu/
• Reasoners:
  – Jena: http://jena.sourceforge.net/
  – OWLAPI: https://github.com/owlcs/owlapi/wiki
Reasoners

• Check ontologies for consistency, infer subsumption relations, answer DL queries
  – HermiT
  – Pellet
  – Fact++

• OWL provides conformance tests for reasoners

• The reasoners can be used inside Protégé or as APIs inside programs
• Syntax is Manchester OWL Syntax
• Start a reasoner to run the query
• Query consistency, classes or individuals
• Consistency:
  • Individual queries: Returns the set of individuals that satisfy a given description
  • Class queries: Returns the class that is either a superclass, subclass, ancestor, or descendant based on what is asked for
Manchester Syntax

- A conjunction of descriptions
- Class name
- DataProperty
  - *value* individual
  - *some* range
  - *min/max/exactly* YYY
- ObjectProperty
  - *some/only* restriction
  - *value* individual
Individual Query (1)

- **Query 1:**
  - Wine
  - Return: BancroftChardonnay, ChateauChevalBlancStEmilion, ...

- **Query 2:**
  - Wine and hasColor value Red
  - Return: ChateauChevalBlancStEmilion, ChiantiClassico, ...

- **Query 3:**
  - Wine and locatedIn value FrenchRegion
  - Return: ChateauChevalBlancStEmilion,
Individual Query (2)

• Query 4:
  • Wine and locatedIn some (Region and adjacentRegion value MendocinoRegion)
  • Wine from Regions that are adjacent to MendocinoRegion
  • Return: CotturiZinfandel, GaryFarrelMerlot, ...

• Query 5:
  – CaliforniaWine and not (hasBody value Medium)
  – Open world assumption
  – Return: CotturiZinfandel, ElyseZinfandel, ...
  – Not: Kavaklidere
Axioms: 1

• Assertions that are given to be true
• Can be especially powerful in combination with other axioms, which may come from different documents
• Some primitives
  – rdfs:subClassOf
  – owl:equivalentClass
Axioms: 2

<owl:AllDifferent> <!-- in essence, pair-wise inequalities→
  <owl:distinctMembers rdf:parseType='Collection'>
    <ex:Country rdf:ID='Russia'/>
    <ex:Country rdf:ID='India'/>
    <ex:Country rdf:ID='USA'/>
  </owl:distinctMembers/>
</owl:AllDifferent>

<ex:Country rdf:ID='Iran'/>
<ex:Country rdf:ID='Persia'>
  <owl:sameIndividualAs rdf:resource='#Iran'/>
</ex:Country>
Restrictions versus Axioms

• Axioms are global assertions that can be used as the basis for further inference

• Restrictions are constructors
  – When we state that hasFather has a maxCardinality of 1, we are
    • Defining the class of animals who have zero or one fathers: this class may or may not have any instances
    • Not stating that all animals have zero or one fathers

• Often, to achieve the desired effect, we would have to combine restrictions with axioms (such as based on equivalentClass)
Vocabulary Semantics

• A trivial ontology
• Uses simple subclasses and properties
  – Disjointness goes beyond RDF
  – Object properties refine RDF properties; relate two objects

<owl:Class rdf:ID="Mammal">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <owl:disjointWith rdf:resource="#Reptile"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasParent">
  <rdfs:domain rdf:resource="#Animal"/>
  <rdfs:range rdf:resource="#Animal"/>
</owl:ObjectProperty>
Simple Inference

• Given the definition for the property hasParent and

<owl:Thing rdf:ID="Fido">
  <hasParent rdf:resource="#Rover"/>
</owl:Thing>

we can infer that Fido is an Animal

• Given that Fido is a mammal, can we infer that Rover is a mammal?
Expressiveness Limitations: 1

OWL DL cannot express some simple requirements

- **Non-tree models**: because instance variables are implicit in OWL restrictions, OWL cannot express conditions that require that two variables be identified
  - Think of siblings – two people who have the same parents – but in terms of classes
  - Do the same thing with class definitions
Expressiveness Limitations: 2

Specialized properties

• Cannot state that the child of a mammal must be a mammal and so on without
  – Defining new child properties for each class
  – Adding an axiom for each class stating that it is a subClassOf the restriction of hasChild to itself

• Analogous to the problem in a strongly typed object-oriented language without generics
  – You have to typecast the contents of a hash table or linked list
Expressiveness Limitations: 3

• Constraints among individuals
  – Cannot define tall person: class of persons whose height is above a certain threshold (current average)
  – Can define ETHusband: class of persons who have been married to Elizabeth Taylor (constant)

• Cannot capture defeasibility (also known as nonmonotonicity)
  – Birds fly
  – Penguins are birds
  – Penguins don’t fly
Ontology Management

• Descriptions of services are improved through the use of ontologies
  – But how do we make sure the parties involved agree upon and understand the ontologies needed?
• Traditional approach: standardize the ontologies via a formal process
• Emerging approach: be more like the Web; figure out the “correct” ontology via consensus
Standard Ontologies

Standardization is more a sociopolitical than a technical process

• IEEE Standard Upper Ontology
• Common Logic (language and upper-level ontology)
• Process Specification Language
• Space and time ontologies
• Domain-specific ontologies, such as health care, taxation, shipping, ...